UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/883,508	06/19/2001	Jeffrey A. Bedell	53470.003042	8696

21967 7590 06/07/2007

HUNTON & WILLIAMS LLP
INTELLECTUAL PROPERTY DEPARTMENT
1900 K STREET, N.W.
SUITE 1200
WASHINGTON, DC 20006-1109

EXAMINER	
ZHEN, LI B	

ART UNIT	PAPER NUMBER
2194	

MAIL DATE	DELIVERY MODE
06/07/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/883,508
Filing Date: June 19, 2001
Appellant(s): BEDELL ET AL.

Brian M. Buroker (Registration No. 39.125)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed February 6, 2007 appealing from the Office action mailed July 26, 2006. It appears that the application header (Application

Number and Docket No.) on pages 2 – 23 have been inadvertently identified as 10/175,328. Since the cover sheet properly identifies the application number and the content of the brief is consistent with the instance application, it appears that this brief is properly submitted for Application No. 09/883,508.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

5,854,932	Mariani et al.	12-1998
6,167,563	Fontana et al.	12-2000
6,112,024	Almond et al.	08-2000

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

- Claims 1, 3 – 5, 10 – 12 and 16 – 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent NO. 5,854,932 to Mariani in view of U.S. Patent No. 6,167,563 to Fontana et al.
- Claims 2, 6 – 9 and 13 – 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mariani and Fontana further in view of U.S. Patent NO. 6,112,024 to Almond.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1, 3 – 5, 10 – 12 and 16 – 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent NO. 5,854,932 to Mariani in view of U.S. Patent No. 6,167,563 to Fontana et al. [hereinafter referred to as Fontana, both references cited in the previous office action].

As to claim 1, Mariani teaches the invention substantially as claimed including a computer implemented method for managing groups of objects [source code files 60 and header files] for use in a reporting system project [a development environment 52 which includes various tools 54 for creating, editing, and compiling source code files 60 and header files 62 for a user's programming project; col. 7, line 60 – col. 8, line 7] comprising the steps of:

receiving a command [modify] to perform a selected function on a selected object [With the editors 70, the user can directly modify or add C++ statements to the source code files 60 and header files 62; col. 8, lines 7 – 22];

identifying using a computer processor dependent objects referred to by the selected object [for each of the object code files 82...the minimal rebuild system 100 generates dependency information...consisting of a set of classes...on which the

respective object code file depends, and a set of dependency types...of the object code file on each class in the set; col. 12, lines 27 – 67];

determining using a computer processor an appropriate manner of executing the selected function [minimal rebuild system 100 determines when recompiling can be avoided by determining how the object code files 82 are dependent on the header files 62; col. 9, lines 1 – 15];

determining using a computer processor appropriate functions to be performed on the dependent objects [minimal rebuild system 100 selectively recompiles the source code files 60...so as to detect changes to all header files 62...that were changed since the last build of the project. The minimal rebuild system 100 then utilizes the detected changes to the header files 62 to determine which of the remaining source code files 60 to recompile and which recompiles can be avoided; col. 17, lines 18 – 30];

causing the appropriate functions to be performed on the dependent objects [omit compiling selected source file and resave object file, step 174, Fig. 6B; compile selected source file into object file, step 176, Fig. 6B; col. 19, lines 33 – 67]; and

causing the execution of the selected function on the selected object in the appropriate manner [minimal rebuild system 100 first recompiles the source code files that were changed since the user's project was last built; col. 17, lines 30 – 43]. Although Mariani teaches the invention substantially as claimed, Mariani does not specifically teach automatically identifying dependent objects, automatically causing appropriate functions to be performed on the dependent objects and automatically causing execution of the selected function on the selected object.

However, Fontana teaches a wrapper tool to develop interfaces for a component or any third party developed software that helps in the component build process [col. 4, lines 10 – 30], automatically identifying dependent objects [dependencies are then analyzed to identify dependent components (block 65); col. 7, lines 22 – 34], automatically causing appropriate functions to be performed on the dependent objects [if the user does want to update dependent components (yes leg of the diamond 66), then the dependent components are retrieved from the source control program (block 69); col. 7, lines 35 – 43] and automatically causing execution of the selected function on the selected object [dependent components are accordingly updated (block 76); col. 7, lines 43 – 58].

It would have been obvious to a person of ordinary skill in the art at the time of the invention to apply the teaching of automatically identifying dependent objects, automatically causing appropriate functions to be performed on the dependent objects and automatically causing execution of the selected function on the selected object as taught by Fontana to the invention of Mariani because this provides a method for building or modifying software components inside a computer system and updating all dependent components automatically in a manner transparent to the user and the computer system [col. 1, lines 33 – 37 of Fontana].

As to claim 10, this is a system claim that corresponds to method claim 1; note the rejection to claim 1 above, which also meet this system claim.

As to claim 18, this is a product claim that corresponds to method claim 1; note the rejection to claim 1 above, which also meet this product claim.

As to claim 3, Mariani teaches the step of causing the appropriate functions to be performed on the dependent objects is performed prior to the step of causing the execution of the selected function in the appropriate manner [minimal rebuild system 100 reorders this list to place any source code files 60 that have changed since the user's project was last rebuilt first in the list; col. 17, lines 43 – 65].

As to claim 4, Mariani teaches the objects are grouped in projects and the selected function relates to manipulating objects within and between projects [For use in creating and editing the source code files 60 and header files 62 of the user's project, the development tools 54 of the environment 52 comprise one or more editors 70, and automated source code generators 72; col. 8, lines 7 – 22] and wherein within each project each object has a unique identifier [search for a name in the scope of the class; col. 13, lines 25 – 50] and a version identifier [date and time of the last change made to the header files; col. 10, lines 8 – 45].

As to claim 5, Mariani teaches comparing dependent objects at a source and objects at a destination to determine whether an object at the destination exists in an identical form to each of the dependent objects at the source and whether an object at the destination exists in a modified form to each of the dependent objects at the source

[minimal rebuild system 100 preferably determines which of the header files 62 were changed since the project was last built by comparing the date stamps of the header files to their last known date stamps; col. 18, lines 34 – 48] and determining, based on the step of comparing, which dependent object to copy from the source to the destination such that the selected object remains complete after execution of the selected function in the appropriate manner [the minimal rebuild system 100 selects the next source code file in the list that is not yet recompiled. At step 171, the minimal rebuild system 100 checks whether the object code file that was compiled in a previous build of the project from the selected source code file still exists; col. 19, lines 7 – 32].

As to claim 11, Mariani teaches the operational module interfaces [classes for interfacing with database management systems; col. 8, lines 23 – 43] with projects that reside in various environments [Class dependency information is similarly reduced for manually generated projects that use a class library; col. 12, lines 28 – 67].

As to claim 12, this is rejected for the same reasons as claim 3 above.

As to claims 16 and 17, these are rejected for the same reasons as claim 5 above.

Claims 2, 6 – 9 and 13 – 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mariani and Fontana further in view of U.S. Patent NO. 6,112,024 to Almond [cited in the previous office action].

As to claim 2, Mariani as modified does not teach the selected object is contained in metadata of an on-line analytical processing system.

However, Almond teaches an object cycle versioning system [col. 2, lines 39 – 52] and an object contained in metadata [maps an object into a meta model which facilitates version control; col. 3, lines 1 – 54 and col. 6, lines 40 – 67] of an on-line analytical processing program [View reports--quickly view project activity status; col. 39, lines 15 – 39].

It would have been obvious to a person of ordinarily skilled in the art at the time of the invention to apply the teaching of storing an object in metadata of an on-line analytical processing system as taught by Almond to the invention of Mariani as modified because this map objects to representations and allow operations supported by the system for versioning will execute correctly even if the objects are stored in a format other than a relational database, such as an object-oriented database, a file server, or other storage system [col. 3, lines 3 – 10 of Almond].

As to claim 6, Mariani as modified teaches the step of receiving is a step of receiving a command to copy a selected object from a source project to a destination

project [Get--copy one or multiple objects to the user's local directory; col. 39, lines 16 – 39 of Almond].

As to claim 7, Mariani as modified teaches the unique identifier [search for a name in the scope of the class; col. 13, lines 25 – 50 of Mariani] and version identifier [date and time of the last change made to the header files; col. 10, lines 8 – 45 of Mariani] of objects in the source project are similar to the unique identifier and version identifier of objects in the destination project [minimal rebuild system 100 preferably determines which of the header files 62 were changed since the project was last built by comparing the date stamps of the header files to their last known date stamps; col. 18, lines 34 – 48 of Mariani].

As to claim 8, Mariani as modified teaches comparing unique identifiers [search for a name in the scope of the class; col. 13, lines 25 – 50 of Mariani] and version identifiers [date and time of the last change made to the header files; col. 10, lines 8 – 45 of Mariani], whether the selected object exists in the destination project in an identical form and whether the selected object exists in the destination project in a modified form [minimal rebuild system 100 preferably determines which of the header files 62 were changed since the project was last built by comparing the date stamps of the header files to their last known date stamps; col. 18, lines 34 – 48 of Mariani]; and

selecting whether to copy the selected object from the source project to the destination project [Get--copy one or multiple objects to the user's local directory; col.

39, lines 16 – 39 of Almond], to replace an object in the destination project with the selected object [compile selected source file into object file, step 176, Fig. 6B; col. 19, lines 33 – 67 of Mariani], and to keep an object in the destination project as is [omit compiling selected source file and resave object file, step 174, Fig. 6B; col. 19, lines 33 – 67 of Mariani].

As to claim 9, Mariani as modified teaches the step of selecting includes prompting the user for a selection [user can directly modify or add C++ statements to the source code files 60 and header files 62; col. 8, lines 7 – 23 of Mariani].

As to claim 13, Mariani as modified teaches the operation module interfaces [RPC interface allows the system to surface an Object Cycle API...for development system clients; col. 2, lines 39 – 54 of Almond] with projects of an on-line analytical processing system [View reports--quickly view project activity status; col. 39, lines 15 – 39 of Almond].

As to claim 14, Mariani as modified teaches the operational module, upon receiving a user command to copy the selected object from a source project to a destination project [Get--copy one or multiple objects to the user's local directory; col. 39, lines 16 – 39 of Almond], determines, by comparing the unique identifiers [search for a name in the scope of the class; col. 13, lines 25 – 50 of Mariani] and the version identifiers [date and time of the last change made to the header files; col. 10, lines 8 –

45 of Mariani], whether the selected object exists in the destination project in an identical form and whether the selected object exists in the destination project in a modified form [minimal rebuild system 100 preferably determines which of the header files 62 were changed since the project was last built by comparing the date stamps of the header files to their last known date stamps; col. 18, lines 34 – 48 of Mariani].

As to claim 15, Mariani as modified teaches the operational module communicates with the user interface to select whether to copy the selected object from the source project to the destination project [Get--copy one or multiple objects to the user's local directory; col. 39, lines 16 – 39 of Almond], to replace an object in the destination object with the selected object [compile selected source file into object file, step 176, Fig. 6B; col. 19, lines 33 – 67 of Mariani], and to keep an object in the destination project as is [omit compiling selected source file and resave object file, step 174, Fig. 6B; col. 19, lines 33 – 67 of Mariani].

(10) Response to Argument

Appellant argues in substance that:

(1) Mariani does not teach or suggest "determining using a computer processor an appropriate manner of executing the selected function" as recited in limitation (iii) of claim 1. The word "manner" refers to the way that something is done, not the fact of whether or not it is done [p. 6];

(2) The Final Office Action, refers to object files, header files, changed files, dependent files and the executable program and asserts that a comparison of dependencies and changes to header files discloses "determining using a computer processor an appropriate manner of executing the selected function on the selected object." Appellant notes that the actions proposed by the Office to allegedly disclose the claim limitation would be redundant with other limitations in the claims. "Identifying dependent objects referred to by the selected object" is recited in the second limitation of claim 1. Thus "determining how the object code files are dependent on the header files" can not disclose "determining using a computer processor an appropriate manner of executing the selected function on the selected object," as recited in the third limitation of claim 1 [pp. 7 – 8].

(3) Furthermore, "detecting changes to the header files" is shown in the context of the cited passage above to be used by the minimal rebuild system of Mariani to avoid recompiling "where any changes to the header files 62 do not affect the resulting object code files 82." (Mariani, col. 9, lines 3-6). Avoiding recompiling is at best a determination of whether or not to execute a selected function and not "an appropriate manner of executing the selected function on the selected object." [p. 8];

(4) Office Action of February 3, 2006 did not show, and the Final Rejection also fails to show a single, coherent embodiment of Mariani for all of the elements for which it is cited. Rather, the Office Action applies disparate features of Mariani that accomplish dissociated tasks. For instance, as to the "selected function" recitation of claim 1, the Office Action alternately applies "modifying code" in element (i) and "recompiling" in

elements (iii) and (vi). Similarly, as to a "selected object," the Office Action appears to apply a "source code file" in element (i), an "object code file" in element (ii), and specific source code files that were changed since the last project build in element (vi). [pp. 8 – 9];

(5) Appellant respectfully submits that updating dependent components in response to a user prompt does not satisfy the "automatically identifying dependent objects," "automatically causing the appropriate functions to be performed on the dependent objects" and "automatically causing the execution of the selected function on the selected object in the appropriate manner" recitations. Requiring user input to automatically update dependent components is not the same as "automatically causing appropriate functions to be performed on the dependent objects." [pp. 9-10];

(6) Updating dependent components automatically is not "automatically causing the appropriate functions to be performed on the dependent objects" [p. 11];

(7) "managing objects with and between projects of a reporting system," has not been addressed. There is no disclosure of managing objects between projects of a reporting system in Mariani [pp. 12-13];

(8) There is no disclosure of a system "wherein the operational module interfaces with projects that reside in various environments," as required by claim 11. The disclosure of a development environment is not the same as the disclosure of a system that interfaces with projects residing in various environments. [p. 13];

(9) The meta model disclosed by Almond does not teach "the selected object is contained in metadata of an on-line analytical processing system." Version control is

not the same as metadata of an on-line analytical processing system. For at least these additional reasons, the rejection of claim 2 should be overturned. [p. 14];

(10) While the Mariani system is specifically designed to selectively recompile object code, it is not configured to copy objects in the manner by which it recompiles object code. Copying and recompiling are different operations that require different systems and methods. Therefore, even if Mariani taught the limitations of claim 1 (which it does not), the Mariani system cannot be combined with Almond to copy objects in the way that it selectively recompiles object code. [pp. 15 and 16];

(11) A reference which does not even contain the term "version control" certainly does not disclose a method or system wherein "wherein the unique identifier and a version identifier of objects in a source project that are similar to the unique identifier and a version identifier of objects in the destination project. [pp. 12 – 13 and 15 – 16];

Examiner respectfully traverses Appellant's arguments:

As to argument (1), Mariani teaches determining using a computer processor an appropriate manner of executing the selected function. Mariani's rebuilding of the executable program through recompiling of the object files corresponds to the claimed "selected function." The minimal rebuild system doesn't just determine whether to execute the function or not. The recompiling function determines how the object code files are dependent on the header files and detecting changes to the header files [col. 9, lines 1 – 15 of Mariani]. Based on these results, the recompiling function determines which files to recompile [the object code files that have changed and the dependent

files]. By determining the changed files and its dependent files, the recompile function determines a particular manner of rebuilding of the executable program. Based on the determination, the rebuilding of the executable program is executed in one of two manners: compile selected source file into object file [step 176, Fig. 6B; col. 19, lines 33 – 67] or omit compiling selected source file and resave object file [step 174, Fig. 6B; col. 19, lines 33 – 67]. Therefore, Mariani teaches determining using a computer processor an appropriate manner [compile selected source file or resave object file] of executing the selected function [rebuilding of the executable program].

As to argument (2), the Final Office Action [response to argument (1), last sentence] refers to the comparison of each object code file's dependencies with the detected changes as disclosing the claimed limitation "determining using a computer processor an appropriate manner of executing the selected function on the selected object". Based on the object code file's dependencies with the detected changes, the minimal rebuild system executes the rebuild function in an appropriate manner [compile selected source file or resave object file]. Examiner also notes that the assertions made in the Final Office Action are not redundant with other limitations in the claims. The second limitation of claim 1 is drawn to the identifying the dependent objects of the selected objects and the corresponding portions of Mariani discloses generation of dependency information for each object code files. For example, step 112, Fig. 4 "dependency analysis" teaches the second limitation of claim 1 and step 116, Fig. 4 "comparison of dependencies to changes" teaches the third limitation of claim 1 [col. 8, line 64 - col. 9, line 15 of Mariani].

As to argument (3), examiner notes that "detecting changes to the header files" is used by the minimal rebuild system to reduce rebuilding of the executable program [col. 8, line 64 – col. 9, line 15]. The rebuilding of the executable program is executed in either one of two manners [compile selected source file or omit compiling selected source file and resave object file] based on comparison of dependencies to changes. Compiling the source file is one manner of executing the selected function [rebuilding the executable program] and omitting compiling of the selected source file and resaving object file is another manner of executing the selected function. In the second manner of rebuilding the executable program, Mariani does not merely disclose omitting compiling of the selected source file, Mariani also discloses resaving the object file.

As to argument (4), examiner notes that modifying corresponds to the user command and the rebuilding corresponds to the selected function. When the user enters the command to modify the source code and header files, the user also invokes the rebuilding function by the minimal rebuild system in response to modification of the source code files and header files [col. 10, lines 8 – 45]. Therefore, the rebuilding function is part of the modification command. The source code and header files correspond to the selected object and the object code files correspond to the dependent object. For example, the object code files in element (ii) corresponds to the dependent objects that are dependent on the selected object [header files; col. 12, lines 57 – 60]. Therefore, the Final Rejections provides coherent embodiments of Mariani for all of the recited elements.

As to argument (5), examiner notes that the Final Rejection did not suggest that updating dependent components in response to a user prompt satisfies the "automatically identifying dependent objects," "automatically causing the appropriate functions to be performed on the dependent objects" and "automatically causing the execution of the selected function on the selected object in the appropriate manner" limitations. The Final office action mapped the limitation "automatically identifying dependent objects" to "dependencies are then analyzed to identify dependent components (block 65); col. 7, lines 22 – 34 of Fontana", "automatically causing the appropriate functions to be performed on the dependent objects" to "if the user does want to update dependent components (yes leg of the diamond 66), then the dependent components are retrieved from the source control program (block 69); col. 7, lines 35 – 43 of Fontana", and "automatically causing execution of the selected function on the selected object" to "dependent components are accordingly updated (block 76); col. 7, lines 43 – 58 of Fontana". Automatically performing a function is interpreted as performing the function with a machine. Fontana discloses that the methods of the invention take the form of program code that is loaded into and executed by a machine [col. 8, lines 60 – 67]. Therefore, the functions in Fontana's invention [i.e. identifying dependent components, retrieving dependent components, and updating the dependent components] are automatically performed by the machine that is executing the program code. Examiner submits that automatically performing a function does not preclude user initiation as long as the functions are performed automatically by the machine. For example, the actual process of retrieving dependent components from the source

control program is performed by the machine and does not require additional instructions from the user. The user initiates execution; however, the step of retrieving dependent components is performed automatically by the machine [i.e. a computer with a processor] executing the instructions.

As to argument (6), the limitation requires automatically performing appropriate functions on the dependent objects [emphasis added]. The limitation "appropriate functions" is very broad and the claims fail to specify the appropriate functions. Therefore, any functions that are performed during execution of the program can be interpreted as appropriate functions. Appellant also fails to provides reasons for the assertion that updating dependent components automatically is not "automatically causing the appropriate functions to be performed on the dependent objects". Examiner submits that updating reads on the broadly recited "appropriate functions". When a file changes, it would be appropriate to update the objects that depend from the file.

In response to argument (7), the recitation "managing objects with and between projects of a reporting system," has not been given patentable weight because the recitation occurs in the preamble. A preamble is generally not accorded any patentable weight where it merely recites the purpose of a process or the intended use of a structure, and where the body of the claim does not depend on the preamble for completeness but, instead, the process steps or structural limitations are able to stand alone. Independent claims 1 and 10 identify the reporting system as the purpose of a process [claim 1: "method for managing groups of object for use in a reporting system"] intended use of a structure [claim 10: "system application for managing objects within

and between projects of a reporting system"]. Examiner submits that the body of claims 1 and 10 are drawn to the execution of a command on a selected object and performing the appropriate functions on the object and its dependent objects and does not depend on the preamble for completeness. Claim 18 only discloses managing groups of objects and does not recite "managing objects with and between projects of a reporting system". Even if the body of the claims depends on the preamble (which it does not), Mariani teaches the limitation "managing objects with and between projects of a reporting system". For example, Mariani discloses performing dependency analysis for projects that use a class library [col. 12, lines 26 – 67]. Mariani manages objects with and between projects when dependency analysis is performed for the projects and it is determined that a class from one project is dependent on a class from another project. As to a "reporting system", it is noted that this limitation is very broad and can be interpreted to be any system that provides information. Mariani teaches a class wizard that reports [provides notification] to the minimal rebuild system [col. 9, lines 57 – 66].

As to argument (8), Mariani discloses interfacing with projects from various environments. Mariani discloses a development environment that provides access to third party code libraries [i.e. Microsoft Foundation Classes, or Object Linking and Embedding; col. 8, lines 22 – 43]. The code libraries are from various environments and the development environment interfaces with a collection of code libraries.

As to argument (9), the Final Rejection mapped "meta model" to the recited metadata. The meta model is mapped from an object and facilitates version control. The meta model provides information about the object that it is mapped from and reads

on the recited metadata. Appellant attempts to define the meaning of metadata by referring to examples of metadata in p. 12, lines 19-21 and p. 13, lines 1-3 of the specification. However, it is noted that the claims do not specifically require these features. In response to applicant's argument that the references fail to show metadata in the context of p. 12, lines 19 – 21 and p. 13, line 1 – 3 of the specification, it is noted that the features upon which applicant relies (i.e., various examples of metadata) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims.

As to argument (10), Mariani discloses rebuilding an execution program in either one of two manners: compiling the source file or resaving an object file. When the minimal rebuild system compiles the source file, it creates a new copy of the object file and when the minimal rebuild system resaves the object file, it makes a copy of the current object file. Therefore, Mariani at least suggests a copying function and is combinable with the system of Almond.

As to argument (11), examiner notes that the claims do not require version control and only recites a version identifier. The date and time of the last change made to the header files [col. 10, lines 8 – 45] in Mariani reads on the recited version identifier. For example, two header files with the same name but different date and time of the last change are different versions of the header file. The header file with the later date and time change is the newer version of the header file and the header file with the earlier data and time change is the older version of the header file.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Li B. Zhen

June 4, 2007

Conferees:

Eddie C. Lee

EDDIE C. LEE
SUPERVISORY PATENT EXAMINER

WILLIAM THOMSON
SUPERVISORY PATENT EXAMINER

William D. Thomson

Li B. Zhen